Parallel Communicating Transducers (Algebraic Systems, Formal Languages and Conventional and Unconventional Computation Theory)

AUTHOR(S):

Cojocaru, Liliana

CITATION:

Cojocaru, Liliana. Parallel Communicating Transducers (Algebraic Systems, Formal Languages and Conventional and Unconventional Computation Theory). 数理解析研究所講究録 2004, 1366: 22-33

ISSUE DATE:

2004-04

URL:

http://hdl.handle.net/2433/25358

数理解析研究所講究録 1366 巻 2004 年 22-33

Parallel Communicating Transducers

Liliana Cojocaru
Rovira i Virgili University, Alexandru I. Cuza, University of Iasi.
e-mails: lc.doc@estudiants.urv.es, clee@infoiasi.ro

Abstract

In this paper a new variant of translating devices is presented. We propose a new model based on transducers working in parallel and communicating with each other by request. We focus our attention upon the strategy of communication, by states or by stacks. We are dealing with the idea of applying them in DNA computing, and in natural language processing, to check and to generate grammatical structures. The generative power of the new systems is tackled in the paper, by taking into consideration the generative power of Watson-Crick automata and several examples.

1 Introduction

The main goal of this paper is to describe a new variant of automata system based on two kinds of accepting devices: transducers and parallel communicating systems. We will call them Parallel Communicating Transducer Systems (henceforth PCTS). The reason of bringing them in the literature is the necessity of improving parallel communicating systems with output capabilities, in order to empower them with the freedom of external communication. The mechanism could have many and interesting applications in computational morphology and phonology, in speech recognition, in splicing systems and in DNA computing, as well. Finite-state transducers have been introduced for the first time in [6]. Since then they have increasingly developed and diversified due to their flexibility in representing and generating a large size of structural data, by using time and space optimal memory [1]. They have been applied especially in natural language processing, in fields such as computational morphology and phonology [9], lexical analyzers [10] or speech recognition [17]. They have been extended to algebraic transducers [11] and weighted finite-state transducer [19]. Transducers with predicates and identities represent another kind of very efficient machines, introduced in [20], in order to characterize phonological rules. Watson-Crick transducers have been developed in order to manipulate DNA molecules. They are described in [21]. If a pushdown automaton is improved with an output, the resulting machine is a pushdown transducer. It comes in the literature from [4], [5] and [7], with many other succeeding papers. Parallel communicating automata are systems composed of automata working independently but communicating with each other, under a specified protocol of cooperation, in order to parse an input string. The protocol depends on the automata type. If all the machines are finite automata then they will communicate through states. If the machines are pushdown automata, the protocol will consist of a pushdown memory interchange strategy. They have been defined in [2], [3], [13], [14], [15] and [16], with forerunner related papers [1], [8].

2 Preliminaries

Systems of cooperating automata have been used till now, to model DNA phenomena, artificial intelligence or communication among agents, but no system has been applied to model natural language [2]. The aim of this section is to find a computational model that could be applied for natural language processing (NLP), in a better way. We have only to improve the automata with an output device and to connect them, by following an efficient strategy of communication

[1] See the transducer minimization algorithm from [18].

[2] There are some trends in this respect, but they merely use the Parallel Communicating Grammar Systems, see e.g. [12].

in order to simulate most of the phenomena that appear in morphology or in phonology. The resulting machine will be a parallel communicating finite transducer system (PCFTS), if the machines are finite transducers. It will be a parallel communicating pushdown transducer system (PCPTS) if the automata are pushdown transducers. The idea of applying them in NLP has come up due to the computational power that parallel communicating systems have. Let X^* be the set of words over an alphabet X, always composed of a finite set of letters, λ the empty word, w^R and $|w|$, $w \in X^*$, the reverse and the length of the word w, respectively. We denote by $X^+ = X^* - \{\lambda\}$, by $|X|$, the cardinality of the set X and by $\bar{X}$ the twin language of X, more exactly $\bar{X} = \{\bar{x}, x \in X\}$.

2.1 Parallel Communicating Finite Transducer Systems

Definition 1. A **parallel communicating finite transducers system of degree** n, ($pcft(n)$ for short) is defined as: $\mathcal{T} = (\Sigma, \Delta, T_1, T_2, \ldots, T_n, K)$,
where:

1. Σ is the input alphabet,
2. Δ is the output alphabet,
3. $T_i = (Q_i, \Sigma, \Delta, \delta_i, q_i, F_i), 1 \leq i \leq n$, are finite transducers with the sets of states Q_i, $q_i \in Q_i$ the initial state of the i^{th} transducer, Δ the finite output alphabet, $F_i \subseteq Q_i$ the set of final states and δ_i the transition mapping of T_i, defined as follows:
$$\delta_i : Q_i \times \Sigma^* \longrightarrow \mathcal{P}(Q_i \times \Delta^*).$$
4. $K = \{K_1, K_2, \ldots, K_n\} \subseteq \bigcup_{i=1}^{n} Q_i$, is the set of query states.

If there exists just one $1 \leq i \leq n$, such that $K \subseteq Q_i$, then the system is said to be **centralized**, the master of this system being the component i.
The system is deterministic if all the components $T_1, \ldots, T_n$ are deterministic. It means that for all $1 \leq i \leq n$ the next conditions are fulfilled:

(i.) $|\delta_i(s, a)| \leq 1$ for all s $\in Q_i$ and a $\in \Sigma \cup \{\lambda\}$,
(ii.) $|\delta_i(s, \lambda)| \neq 0$ for some s $\in Q_i$ then $|\delta_i(s, a)| = 0$ for all a $\in \Sigma$.

If the transition mapping δ_i, is defined as: $\delta_i : Q_i \times \Sigma \cup \{\lambda\} \longrightarrow \mathcal{P}(Q_i \times \Delta^*)$, for all i, $1 \leq i \leq n$, then the system is said to be a **parallel communicating sequential transducers of degree** n ($pcst(n)$ for short). By a configuration of a $pcft(n)$ we mean an $3n$-tuple:
$$(s_1, x_1, \gamma_1, s_2, x_2, \gamma_2, \ldots, s_n, x_n, \gamma_n)$$
where:

- s_i is the current state of the component i,
- x_i is the remaining part of the input word which has not yet been read by T_i,
- $\gamma_i \in \Delta^*$ is the output string emitted up to this point by T_i, $1 \leq i \leq n$.

The binary relations, on the set of all configurations of $\mathcal{T}$ are defined as:

(1) $$(s_1, x_1, \gamma_1, \ldots, s_n, x_n, \gamma_n) \vdash (p_1, y_1, \gamma_1 z_1, \ldots, p_n, y_n, \gamma_n z_n)$$
iff one of the following two conditions holds:

(1.i) $K \cap \{s_1, s_2, \ldots, s_n\} = \emptyset$ and $x_i = a_i y_i, a_i \in \Sigma \cup \{\lambda\}, (p_i, z_i) \in \delta_i(s_i, a_i), 1 \leq i \leq n$,
(1.ii) for all $1 \leq i \leq n$ such that $s_i = K_{j_i}$ and $s_{j_i} \notin K$ put $p_i = s_{j_i}, 1 \leq j_i \leq n, p_r = s_r$, for all the other $1 \leq r \leq n$, and $y_t = x_t$, $1 \leq t \leq n$.

(2) $$(s_1, x_1, \gamma_1, \ldots, s_n, x_n, \gamma_n) \vdash_r (p_1, y_1, \gamma_1 z_1, \ldots, p_n, y_n, \gamma_n z_n,)$$
iff one of the following two conditions holds:

(2.i) $K \cap \{s_1, s_2, \ldots, s_n\} = \emptyset$ and $x_i = a_i y_i, a_i \in \Sigma \cup \{\lambda\}, (p_i, z_i) \in \delta_i(s_i, a_i), 1 \leq i \leq n$,
(2.ii) for all $1 \leq i \leq n$ such that $s_i = K_{j_i}$ and $s_{j_i} \notin K$ put $p_i = s_{j_i}, p_{j_i} = q_{j_i}, 1 \leq j_i \leq n$, and $p_r = s_r$ for all the other $1 \leq r \leq n$, and $y_t = x_t$, $1 \leq t \leq n$.

A $pcft(n)$ system whose moves are all based on the relation $\vdash_r$ is said to be **returning** (henceforth $rpcft(n)$), and **non-returning** otherwise.

Definition 2. Let $\mathcal{T} = (\Sigma, \Delta, T_1, T_2, \ldots, T_n, K)$ be a PCFTS, returning or not. We say that a word

$v \in \Delta^*$ **is an output for an input string** $u \in \Sigma^*$, if and only if there exist $u_{i1}, \ldots, u_{ik} \in \Sigma^*$; $v_{i1}, \ldots, v_{ik} \in \Delta^*$, $q_{i0}, \ldots, q_{ik} \in Q_i$[3], where $k \in N^*$, q_{i0} being the initial state of the transducer T_i, such that, for $1 \leq i \leq n$ the next conditions are fulfilled:

i. $u = u_{i1} \ldots u_{ik}$ and $v = v_{i1} \ldots v_{ik}$,

ii. $(q_{ij+1}, v_{ij+1}) \in \delta_i (q_{ij}, u_{ij+1})$, with $q_{ik} \in F_i$, $1 \leq i \leq n$, $0 \leq j < k$;

such that one of the following conditions have to be realized:

ii.1. If $K \cap \{q_{1j}, q_{2j}, \ldots, q_{nj}\} = \emptyset$, then due to 1.*i* or 2.*i*, at the step $j+1$ there will be an usual transition (without any communication);

ii.2. If $K \cap \{q_{1j}, q_{2j}, \ldots, q_{nj}\} \neq \emptyset$, then for all $i \in \{1, 2, \ldots, n\}$ such that $q_{ij} = K_{l_j}$; $l_j \in \{1, 2, \ldots, n\}$, $l_j \neq i$,and $q_{l_j j} \notin K$, using 1.*ii*, **for a non-returning system**, we will have: $q_{ij+1} = q_{l_j j}$. **For a returning system**, applying 2.*ii*, besides the last condition, we will also have: $q_{l_j j+1} = q_{l_j 0}$ (it means that the transducer T_{l_j} gets into the initial state: $q_{l_j 0}$ at the $(j+1)$-th transition).

For short, using the relation defined in (1), an word $v \in \Delta^*$ is an output for $u \in \Sigma^*$, in a $pcft(n)$, if and only if:

$$(3) \qquad (q_1, u, \lambda, \ldots, q_n, u, \lambda) \vdash^* (s_1, \lambda, v, \ldots, s_n, \lambda, v), \qquad s_i \in F_i, 1 \leq i \leq n;$$

where $\vdash^*$ is the reflexive and transitive closure of the relation $\vdash$.

Using the relation (2), $v \in \Delta^*$ is an output for $u \in \Sigma^*$ in a $rpcft(n)$ if:

$$(4) \qquad (q_1, u, \lambda, \ldots, q_n, u, \lambda) \vdash_r^* (s_1, \lambda, v, \ldots, s_n, \lambda, v), \qquad s_i \in F_i, 1 \leq i \leq n;$$

in which $\vdash_r^*$ is the reflexive and transitive closure of the relation $\vdash_r$.

If in definition 2. we consider $u_{i1}, \ldots, u_{ik} \in \Sigma$ then the output word for a $pcst(n)$, having as input the word $u \in \Sigma^*$, is obtained.

Let $\mathcal{T} = (\Sigma, \Delta, T_1, T_2, \ldots, T_n, K)$ be a $pcft(n)$. For a given computation σ, displayed in (3), we define by $\pi(\sigma)$ the sequence of labels of transition rules used in σ. The controlled language of the $pcft(n)$ is defined as:

$$L_{ctr}(\mathcal{T}) = \{\pi(\sigma) | \sigma : (q_1, u, \lambda, \ldots, q_n, u, \lambda) \vdash^* (s_1, \lambda, v, \ldots, s_n, \lambda, v),\ s_i \in F_i,\ 1 \leq i \leq n\},$$

in which v is the output word for $\mathcal{T}$, when the input string v has been read. We denote by PCFT(n)(ctr) the family of controlled languages of $pcft(n)$ over a given alphabet. In the same way the controlled language and the family of the controlled languages for a $pcst(n)$, $rpcft(n)$, or $rpcst(n)$ is defined. If $L \subseteq \Sigma^*$ is a language over Σ, accepted by the associated $pcfa(n)$, then the output language generated by $\mathcal{T}$ (*the PCFT mapping*) is defined as:

$$T(L) = \{v \mid \text{ such that } (u, v) \in \tau(T),\ u \in L\},$$

in which $\tau(T)$ is *the transduction realized by* T, more exactly :

$\tau(\mathbf{T})$={(u, v) |where $u \in \Sigma^*, v \in \Delta^*$ satisfy (3) for a PCFTS or (4) for a RPCFTS}

2.2 Parallel Communicating Pushdown Transducer Systems

Definition 3. A parallel communicating pushdown transducers system of degree n ($pcpt(n)$ for short) is a system: $\mathcal{T} = (\Sigma, \Delta, \Gamma, T_1, T_2, \ldots, T_n, K)$,

where:

1. Σ is the input alphabet ,
2. Δ is the output alphabet ,
3. Γ is the alphabet of pushdown symbols ,
4. for each $1 \leq i \leq n$, $T_i = (Q_i, \Sigma, \Gamma, \Delta, \delta_i, q_i, Z_i, F_i)$ is a pushdown transducer with the set of states Q_i, the initial state $q_i \in Q_i$, the input alphabet Σ, the pushdown alphabet Γ, Δ the finite output alphabet, the initial contents of the pushdown memory $Z_i \in \Gamma$, the set of final states $F_i \subseteq Q_i$, and the transition mapping :
$$\delta_i : Q_i \times (\Sigma \cup \{\lambda\}) \times \Gamma \longrightarrow \mathcal{P}(Q_i \times \Gamma^* \times \Delta^*).$$
5. $K \subseteq \{K_1, K_2, \ldots, K_n\} \subseteq \Gamma$ is the set of query symbols.

A configuration of a $pcpt(n)$ is a $4n$-tuple:

$$(s_1, x_1, \alpha_1, \gamma_1, s_2, x_2, \alpha_2, \gamma_2, \ldots, s_n, x_n, \alpha_n, \gamma_n)$$

where:

[3] $q_{i0}, \ldots, q_{ik}$ not necessarily being distinct

- $s_i \in Q_i$ is the current state of the component T_i,
- $x_i \in \Sigma^*$ is the remaining part of the input word which has not been read yet by T_i,
- $\alpha_i \in \Gamma^*$ is the contents of the i-th stack ,
- $\gamma_i \in \Delta^*$ is the output string emitted up to that point, $1 \leq i \leq n$.

Two kinds of transition relations are defined on the set of all configurations of $\mathcal{T}$, as follows:

(5) $(s_1, x_1, B_1\alpha_1, \gamma_1, \ldots, s_n, x_n, B_n\alpha_n, \gamma_n) \vdash (p_1, y_1, \beta_1\alpha_1, \gamma_1 Z_1, \ldots, p_n, y_n, \beta_n\alpha_n, \gamma_n Z_n)$,

where $B_i \in \Gamma, \alpha_i, \beta_i \in \Gamma^*$ and $\gamma_i, Z_i \in \Delta^*$ for all $1 \leq i \leq n$, iff one of the following two conditions holds, for all $1 \leq i \leq n$:

(5.i) $K \cap \{B_1, B_2, \ldots, B_n\} = \emptyset$ and $x_i = a_i y_i, a_i \in \Sigma \cup \{\lambda\}, (p_i, \beta_i, z_i) \in \delta_i(s_i, a_i, B_i)$,

(5.ii) for all $i, 1 \leq i \leq n$ such that $B_i = K_{j_i}$ and $B_{j_i} \notin K, \beta_i = B_{j_i}\alpha_{j_i}$,

for all other $r, 1 \leq r \leq n, \beta_r = B_r$, and $y_t = x_t$, $p_t = s_t$, for all $t, 1 \leq t \leq n$.

(6) $(s_1, x_1, B_1\alpha_1, \gamma_1, \ldots, s_n, x_n, B_n\alpha_n, \gamma_n) \vdash_r (p_1, y_1, \beta_1\alpha_1, \gamma_1 Z_1, \ldots, p_n, y_n, \beta_n\alpha_n, \gamma_n Z_n)$,

where $B_i \in \Gamma, \alpha_i, \beta_i \in \Gamma^*, \gamma_i, Z_i \in \Delta^*$, $1 \leq i \leq n$, iff one of the following two conditions holds, for all $1 \leq i \leq n$:

(6.i) $K \cap \{B_1, B_2, \ldots, B_n\} = \emptyset$ and $x_i = a_i y_i, a_i \in \Sigma \cup \{\lambda\}, (p_i, \beta_i, z_i) \in \delta_i(s_i, a_i, B_i)$,

(6.ii) for all $1 \leq i \leq n$ such that $B_i = K_{j_i}$ and $B_{j_i} \notin K, \beta_i = B_{j_i}\alpha_{j_i}$, and $\beta_{j_i} = Z_{j_i}$,

for all the other $r, 1 \leq r \leq n, \beta_r = B_r$, and $y_t = x_t$, $p_t = s_t$, for all $t, 1 \leq t \leq n$.

The terminology, such as *deterministic*, *centralized*, or *a returning* PCPTS used for the PCFTS systems, is carried over to these similar devices, too. If the associated pushdown automaton A_i, to the pushdown transducer T_i, is an extended one, for each $1 \leq i \leq n$, then the system $\mathcal{T}$, is said to be an extended PCPTS. It means that the transition mapping δ_i is defined as:

$\delta_i : Q_i \times (\Sigma \cup \{\lambda\}) \times \Gamma^* \longrightarrow \mathcal{P}(Q_i \times \Gamma^* \times \Delta^*)$, for all $1 \leq i \leq n$.

Transitions for an EPCPT will be defined in the same way as for a PCPTS, with difference that extended systems may read words $B_i \in \Gamma^*$ instead of symbols.

Definition 4. Let $\mathcal{T} = (\Sigma, \Delta, \Gamma, T_1, T_2, \ldots, T_n, K)$ be a PCPTS. We say that a word $v \in \Delta^*$ **is an output for the input string** $u \in \Sigma^*$, if and only if there exist $u_{i1}, \ldots, u_{ik} \in \Sigma^*$; $v_{i1}, \ldots, v_{ik} \in \Delta^*$, $q_{i0}, q_{i1}, \ldots, q_{ik} \in Q_i$, for all $1 \leq i \leq n$, such that :

i. $u = u_{i1} \ldots u_{ik}$ and $v = v_{i1} \ldots v_{ik}$,

ii. $(q_{ij+1}, Z_{ij+1}, v_{ij+1}) \in \delta_i(q_{ij}, u_{ij+1}, \gamma_{ij})$ for $0 \leq j < k$ and $1 \leq i \leq n$, where:

ii.1. **If** $K \cap \{\gamma_{1j}, \gamma_{2j}, \ldots, \gamma_{nj}\} = \emptyset$, concerning with 5.*i* or 6.*i*, at the step j+1, it will be an usual transition, no stack communication;

ii.2. **If** $K \cap \{\gamma_{1j}, \gamma_{2j}, \ldots, \gamma_{nj}\} \neq \emptyset$, then for all $i \in \{1, 2, \ldots, n\}$ such that $\gamma_{ij} = K_{l_j}$ and $\gamma_{l_j} \notin K$ using the relations 5.*ii*, **for a non-returning system**, it will be: $Z_{ij+1} = B_{l_j j+1}\alpha_{l_j j+1}$ (it means that, at the transition j+1, the contents of the stack of the transducer T_{l_j}, will be sent to the stack T_i). **For a returning system**, concerning with 6.*ii*, besides the last condition, we will also have: $Z_{l_j j+1} = Z_{l_j 0}$ (it means that at the transition j+1, the contents of the stack of the T_{l_j} transducer, contains only the initial symbol $Z_{l_j 0}$).

For short, using the relation defined in (5), a word $v \in \Delta^*$ is an output for $u \in \Sigma^*$, using a *pcpt(n)*, if and only if:

(7) $(q_1, u, Z_1, \lambda, \ldots, q_n, u, Z_n, \lambda) \vdash^* (s_1, \lambda, \alpha_1, v, \ldots, s_n, \lambda, \alpha_1, v), s_i \in F_i, 1 \leq i \leq n$ for a PCPTS accepting by final states, and :

(7') $(q_1, u, Z_1, \lambda, \ldots, q_n, u, Z_n, \lambda) \vdash^* (s_1, \lambda, \lambda, v, \ldots, s_n, \lambda, \lambda, v), s_i \in Q_i, 1 \leq i \leq n$ for a PCPTS accepting by empty pushdown memory .

Replacing the relation $\vdash^*$ with the relation $\vdash_r^*$ in (7) and in (7'), the definition of an output word, for a RPCPT accepting by final states or by empty pushdown memory is obtained, respectively. In the same way as for PCFTS, the notions of *controlled language of a PCPT*, or *PCPT mapping* can be introduced.

3 The computational power of these systems

The problem that arises now, is to study the type of languages generated by a PCFTS or by a PCPTS, having as input an arbitrary language $L \subseteq \Sigma^*$.

Let *RE, CS, CF, LIN* and *REG* be the families of languages generated by arbitrary, context-sensitive, context-free, linear and regular grammars, respectively. The next inclusion holds:

(8) $$REG \subset LIN \subset CF \subset CS \subset RE$$

For each parallel communicating automata system we can build a PCFTS or a PCPTS, having the same structure with the associated automaton. Following this, they are able to accept and to generate the same kind of languages. The questions arising now are the following:

" *Given an input language from a class X of the hierarchy (8), could it be possible to generate a language from a class Y such that* $X \subset Y$ *?*" or

" *Which is the very first class of languages from the hierarchy (8) that could be used by a PCTS to cover the RE family ?*", and even more:

" *Which is the minimal number of components that a PCTS should have in order to satisfy the above goals?*"

Answers to these questions will be given in the sequel.

3.1 Watson-Crick automata and PCFAS with two components - a computational power analogy

A parallel communicating finite automata system (PCFAS) could be viewed as a PCTS that generates only empty strings. It is because the connection and the communication between the components work exactly in the same manner. That is why we will skip here the definition[4] of these devices. In the following the notations $rpcfa(2)$, $rcpcfa(2)$, $cpcfa(2)$ denote the returning, returning centralized, and centralized PCFAS with two components. If x(2) is a type of the above system, then X(n) is the class of all languages accepted by automata systems of type x. We will show in this subsection that $rpcfa(2)$, $rcpcfa(2)$ and $cpcfa(2)$ are computational equivalent with 1-limited Watson-Crick finite automata (henceforth 1WK). The reason has been suggested by the Theorem 1. and Theorem 2. from [15] in which for n=2, we get to the computational equivalence between $pcfa(2)$ and two-head finite automata. On the other hand it is also known that two-head finite automata have the same power as 1WK [21]. So that, if we denote by 1WK(u) the languages generated by 1WK, the next result holds:

Lemma 1 PCFA(2) = 1WK(u)

It is not clear if the above result holds also for the returning or centralized finite automata systems. Consequently, in the following, we will focus our attention on solving this problem.

Definition 5. A **Watson-Crick finite automaton** (WK) is a construct: $\mathcal{M} = (\Sigma, \rho, Q, s_0, F, \delta)$, where Σ is the input alphabet and Q is the set of states, Σ and Q are disjoint sets, $\rho \subseteq \Sigma \times \Sigma$ is a complementarity relation, $s_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \binom{\Sigma^*}{\Sigma^*} \longrightarrow P(Q)$ is the transition mapping such that $\delta(s, \binom{x}{y}) \neq \phi$ only for finitely many triples $(s, x, y) \in Q \times \Sigma^* \times \Sigma^*$.

We say that $s' \in \delta(s, \binom{x_1}{x_2})$, if in the state s, the automaton passes over x_1, in the upper level strand and over x_2 in the lower level strand of a double stranded sequence, and enters the state s'. This can be written, in terms of rewriting rules as: $s\binom{x_1}{x_2} \longrightarrow \binom{x_1}{x_2}s'$. A transition in a WK automaton is defined as follows:

for $\binom{x_1}{x_2}, \binom{u_1}{u_2}, \binom{w_1}{w_2} \in \binom{\Sigma^*}{\Sigma^*}$ such that $\begin{bmatrix} x_1 u_1 w_1 \\ x_2 u_2 w_2 \end{bmatrix} \in WK_\rho(\Sigma)$ and $s, s' \in Q$, we write:

$$\binom{x_1}{x_2} s \binom{u_1}{u_2} \binom{w_1}{w_2} \Longrightarrow \binom{x_1}{x_2} \binom{u_1}{u_2} s' \binom{w_1}{w_2}.$$

We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$. The language recognized by the upper strands [5] of Watson-Crick tapes is defined as:

$$L_u(M) = \{w_1 \in V^* / s_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Longrightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} s_f, \text{for } s_f \in F, w_2 \in V^*, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)\}.$$

For a computation $\sigma : s_0 w \Longrightarrow^* w s_f, w \in WK_\rho(\Sigma), s_f \in F$, we denote by $e(\sigma) \simeq word of \sigma$, that is the sequence of labels of transition rules used in σ. the controlled language of the WK automaton

[4] The reader can find it in the references.

[5] In general only the language appearing in the upper strand is studded, the other one being linked by the first through the complementarity relation ρ.

over an alphabet Σ, is defined as:

$$L_{ctr}(M) = \{e(\sigma) |\ \sigma : s_0 w \Longrightarrow^* w s_f, w \in \mathrm{WK}_\rho(\Sigma), s_f \in \mathrm{F}\}.$$

Several variants of WK finite automata have been introduced in the literature. Only the next two types are useful in our considerations:

- **stateless**, if in the definition 5. we have $\mathrm{Q} = \mathrm{F} = s_0$;
- **1-limited** if for all $s\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \longrightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} s' \in P$ we have $|x_1 x_2| = 1$.

We denote by $\mathrm{AWK}(\alpha)$, $\mathrm{NWK}(\alpha)$, $\mathrm{1WK}(\alpha)$, $\mathrm{N1WK}(\alpha)$ the family of languages $L_\alpha(M)$, $\alpha \in \{u, ctr\}$ recognized by WK finite automata which are arbitrary, stateless, 1-limited, stateless and 1-limited, respectively. Several results related to the above classes, can be found in [21], besides the next inclusion holds:

(9) $$REG \subset AWK(u) = 1WK(u) \subset CS$$

Following the above statements we claim that:

Theorem 1.
1. $RPCFA(2) = RCPCFA(2) = CPCFA(2) = PCFA(2) = 1WK(u)$.
2. For every family $X(2)$, $X(2) \subset CS$, $X \in \{RPCFA, RCPCFA, CPCFA, PCFA\}$.

In order to prove Theorem 1. we will show that:

S_1. *For each $rpcfa(2)$, $rcpcfa(2)$ or $cpcfa(2)$ one can construct an 1WK that generates the same language*; and vice-versa,

S_2. *For each 1WK there exist a $rpcfa(2)$, a $rcpcfa(2)$ and a $cpcfa(2)$, that can be effectively constructed to simulate the 1WK automaton.*

S_1. Let $\mathcal{A} = (\Sigma, A_1, A_2, K)$ be a $rpcfa(2)$, where $A_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ are finite automata systems. Let $\mathcal{M} = (\Sigma, \rho, Q_1 \cup Q_2, q_0, F, \delta)$ be an WK automaton, in which $q_0 \in \{q_1, q_2\}$ is the initial state, $F \subset F_1 \cup F_2$ the set of final states and the complementarity relation being the identity, $(a, b) \in \rho$ if and only if a = b. Let w be an arbitrary word accepted by $\mathcal{A}$. This word will be decomposed by A_1 and A_2, in two different partitions as follows: $w = a_{11}a_{12}a_{13}...a_{1k} = a_{21}a_{22}a_{23}...a_{2k}$ where $a_{ij} \in \Sigma \cup \{\lambda\}$, and $(q_1, a_{11}a_{12}a_{13}...a_{1k}, q_2, a_{21}a_{22}a_{23}...a_{2k}) \vdash^k (s_1, \lambda, s_2, \lambda)$.

We have to define the δ transitions for the WK automaton such that at each odd transition the 1WK reads in the upper strand the symbol that has been read by the first automaton, and at each even transition the 1WK reads in the lower strand the symbol that has been read by the second automaton. Suppose that at the step j, $1 \leq j < k$, $rpcfa(2)$ performs the transition:

(10) $$(q_{1j-1}, a_{1j}a_{1j+1}...a_{1k}, q_{2j-1}, a_{2j}a_{2j+1}...a_{2k}) \vdash (q_{1j}, a_{1j+1}...a_{1k}, q_{2j}, a_{2j+1}...a_{2k})$$

This will be simulated by the 1WK, as follows:

$$q_{1j-1}\binom{a_{1j}...a_{1k}}{a_{2j}...a_{2k}} \vdash \binom{a_{1j}}{\lambda} q_{2j-1}\binom{a_{1j+1}...a_{1k}}{a_{2j}...a_{2k}} \vdash \binom{a_{1j}}{a_{2j}} q_{1j}\binom{a_{1j+1}...a_{1k}}{a_{2j+1}...a_{2k}} \vdash \binom{a_{1j}a_{1j+1}}{a_{2j}} q_{2j}\binom{a_{1j+2}...a_{1k}}{a_{2j+1}...a_{2k}}$$

For each q_{1j-1}, q_{2j-1}, q_{1j} and q_{2j} that do not belong to the set of queries K, the δ mapping has to be defined as: $\delta(q_{1j-1}, \binom{a_{1j}}{\lambda}) = q_{2j-1}$ (derivation applied at each odd step)

$\delta(q_{2j-1}, \binom{\lambda}{a_{2j}}) = q_{1j}$ (derivation applied at each even step).

If $q_{1j-1} = K_2$ then, using the definition of the RPCFA, see e.g. [15], (10) becomes :

(11) $$(q_{2j-1}, a_{1j}a_{1j+1}...a_{1k}, q_2, a_{2j}a_{2j+1}...a_{2k}) \vdash (q_{1j}, a_{1j+1}...a_{1k}, q_{2j}, a_{2j+1}...a_{2k})$$

so that δ has to be defined as: $\delta(q_{2j-1}, \binom{a_{1j}}{\lambda}) = q_2$, $\delta(q_2, \binom{\lambda}{a_{2j}}) = q_{1j}$.

If in (11) we have $q_{1j} = K_2$ then at the next step we have to define δ as:

$$\delta(q_{2j-1}, \binom{a_{1j}}{\lambda}) = q_2, \quad \delta(q_2, \binom{\lambda}{a_{2j}}) = q_{2j} \quad \text{and} \quad \delta(q_{2j}, \binom{a_{1j+1}}{\lambda}) = q_2.$$

If $q_{2j-1} = K_1$ then the sequence (10) becomes :

(12) $$(q_1, a_{1j}a_{1j+1}...a_{1k}, q_{1j-1}, a_{2j}a_{2j+1}...a_{2k}) \vdash (q_{1j}, a_{1j+1}...a_{1k}, q_{2j}, a_{2j+1}...a_{2k})$$

so that the mapping δ will be: $\delta(q_1, \binom{a_{1j}}{\lambda}) = q_{1,j-1}$, $\delta(q_{1j-1}, \binom{\lambda}{a_{2j}}) = q_{1j}$, and so on.

To notice that because the symbols $(a_{ij})_{i \in \{1,2\}, j \in \{1..k\}}$ have been chosen in $\Sigma \cup \{\lambda\}$, it is possible pairs such as $\binom{a_{1j}}{\lambda}$ or $\binom{\lambda}{a_{2j}}$ to be equal with $\binom{\lambda}{\lambda}$. These situations can be eliminated. For instance, if in (12) we have $a_{2j} = \lambda$, and $q_{2j} = K_1$ then δ have to be defined as: $\delta(q_1, \binom{a_{1j}}{\lambda}) = q_1$, and $\delta(q_1, \binom{a_{1j+1}}{\lambda}) = q_{1j}$ such that at the end of the computation the automaton will be a pure 1WK. Because the transition δ has been constructed following step by step the automaton transitions that generate a specified language, the 1WK automaton will generate the same language. It is

well known from [15], that each *rcpcfa*(2) can be simulated by a *rpcfa*(2) and consequently each *rcpcfa*(2) can be simulated by an 1WK automaton. In the same paper it is shown that each *cpcfa*(2) can be converted into a *pcfa*(2) and due to Lemma1, each *cpcfa*(2) can be simulated by an 1WK. With these remarks the statement S_1 has been proved.

S_2. Vice-versa, let $\mathcal{M}=(\Sigma,\rho,Q,s_0,F,\delta)$ be an 1WK automaton, that generates the language $L_u(\mathcal{M})$, and w an arbitary word from $L_u(\mathcal{M})$. There exist a derivation such that $s_1\binom{w}{\bar{w}}\Longrightarrow^*\binom{w}{\bar{w}}s_f$. Let $w=a_1a_2...a_k$ and $\bar{w}=\bar{a}_1\bar{a}_2...\bar{a}_k$ be the decomposition of w and $\bar{w}$ respectively, by following the 1WK derivation steps, where $a_i,\ \bar{a}_i\in\Sigma\cup\{\lambda\}$. Suppose that in step j, $1\leq j\leq k$ the automaton performs the transitions:

$$q_j\binom{a_ja_{j+1}...a_k}{\bar{a}_j\bar{a}_{j+1}...\bar{a}_k}\vdash\binom{a_j}{\bar{a}_j}q_{j+1}\binom{a_{j+1}...a_k}{\bar{a}_{j+1}...\bar{a}_k}\vdash\binom{a_ja_{j+1}}{\bar{a}_j\bar{a}_{j+1}}q_{j+2}\binom{a_{j+2}...a_k}{\bar{a}_{j+2}...\bar{a}_k}\vdash ... \tag{13}$$

where $|a_l\bar{a}_l|=1$, and $q_l\in Q$ for each $j\leq l\leq k$. In terms of δ rules the above transitions are defined as:

$$\delta(q_j,\binom{a_j}{\bar{a}_j})=q_{j+1},\quad \delta(q_{j+1},\binom{a_{j+1}}{\bar{a}_{j+1}})=q_{j+2} \tag{14}$$

Let $\mathcal{A}=(\Sigma,A_1,A_2,K)$ be the *rpcfa*(2) that we want to develop. Let s_0 be the initial state of the first automaton, r_0 the initial state of the second automaton, r_0 not being necessarily from Q, $Q_1\subseteq Q$ and $Q_2\subseteq\{r_0\}\cup Q$, $F_1=F_2=F$ the sets of final states and the mappings δ_i, $i\in\{1,2\}$, defined below. At each derivation step done by the 1WK automaton the first automaton of the PCFAS reads the string from the upper strand, the second automaton reads the string from the lower strand, and the current state of the 1WK has to be taken over by one of the PCFAS components. For a returning, non-centralized system, the strand that contains λ determines which one of the components has to communicate, a component will be in the current state of the 1WK automaton and the other one will be in its initial state. For a centralized type only the chosen master has to ask, both automata being in the current 1WK state for a non-returning system. To note also that each δ rule from the 1WK will be simulated by two rules in a PCFAS, that depend on the last pair of states in which the 1WK automaton has been stated. That is why in the construction of δ_i, $i\in\{1,2\}$ we have to follow step by step, the 1WK derivations, in order not to miss the same δ rule but in a different pair of states. Let us suppose that at the transition step j-1 the system finishes in the pair of states (q_j,r_0). Considering the above remarks, the first rule from (14) will be rewritten as follows:

If $a_j=\lambda$ then $\delta(q_j,\binom{\lambda}{\bar{a}_j})=q_{j+1}$ becomes: $\delta_1(q_j,\lambda)=K_2$ (because of the empty string in the upper strand) and $\delta_2(r_0,\bar{a}_j)=q_{j+1}$ (because one of the automata has to conceive the 1WK state) and, because of the returning condition, the *rpcfa*(2) system will finish this step in the pair of states: (q_{j+1},r_0).

If, in (14), $\bar{a}_j=\lambda$ then $\delta(q_j,\binom{a_j}{\lambda})=q_{j+1}$ will be simulated by $\delta_1(q_j,a_j)=q_{j+1}$ and $\delta_2(r_0,\lambda)=K_1$, after which the system gets into the states: (s_0,q_{j+1}).

Because the transitions δ_i, $i\in\{1,2\}$ have to be constructed following step by step the 1WK transitions, that generate a specified language, denoted by $L_u(\mathcal{M})$, the *rpcfs*(2) system will generate the same language [6]. We conclude that the above constructed *rpcfa*(2) simulates the 1WK automaton, and it had been correctly defined. In order to obtain a *rcpcfa*(2) we have to allow only one component to ask. Supposing that the second component has this permission, and at the step j-1, the system gets into the states: (s_0,q_j) then, at the next step j, the δ function will be rewritten in: $\delta_1(s_0,\lambda)=q_{j+1}$ and $\delta_2(q_j,\bar{a}_j)=K_1$, if $a_j=\lambda$. It will be transformed in $\delta_1(s_0,a_j)=q_{j+1}$ and $\delta_2(q_j,\lambda)=K_1$, if $\bar{a}_j=\lambda$. In all the cases the system finishes the communication in the states (s_0,q_{j+1}). Because of the centralized character of the system only the second automaton has to conceive the 1WK current state, the first one finishing always in its initial state. If we want to convert the 1WK automaton into a *cpcfa*(2) then we have to allow only one of the automata to ask without letting the other one to return in its initial state. Such an automaton will finish the transition step j-1 in the states (q_j,q_j), so that the first function, from (14) will be rewritten

[6] Nothing had been changed, we had only built the transitions δ_1 and δ_2 and synchronized them with respect, to the RPCFA definition.

in: $\delta_1(q_j,\lambda)=q_{j+1}$ and $\delta_2(q_j,\bar{a}_j)=K_1$, if $a_j=\lambda$ or in $\delta_1(q_j,a_j)=q_{j+1}$ and $\delta_2(q_j,\lambda)=K_1$, if $\bar{a}_j=\lambda$.
If we denote by $L_{ctr}(A)$ the controlled language defined by A over an alphabet Σ, that is the set of sequences of labeled transition rules for an accepting computation, and by $PCFA(2)(ctr)$ the family of controlled languages for a $pcfa(2)$, then the next result is an immediate consequence of the theorem 1:

Corollary 1 $1WK(ctr)=PCFA(2)(ctr)$

In the next example we show how to build, using the above explained method, a $rpcfa(2)$, a $rcpcfa(2)$ or a $cpcfa(2)$ starting from an 1WK automaton that accepts the language L = $\{a^nb^nc^n|n\geq 1\}$.

Example 1. Let $\mathcal{M}=(\{a,b,c\},\rho,\{s_a,s_b,s_c\},s_a,F=\{s_a\},\delta)$, be the 1WK automaton that accepts the language L, in which $\rho=\{(x,x)|x\in\Sigma\}$, and the function δ is defined as:

1. $\delta(s_a,\binom{a}{\lambda})=s_a$, 2. $\delta(s_a,\binom{b}{\lambda})=s_b$, 3. $\delta(s_b,\binom{\lambda}{a})=s_a$,
4. $\delta(s_a,\binom{c}{\lambda})=s_c$, 5. $\delta(s_c,\binom{\lambda}{b})=s_a$, 6. $\delta(s_a,\binom{\lambda}{c})=s_a$.

the controlled language of $\mathcal{M}$ over $\Sigma=\{a,b,c\}$ is:

$$L_{ctr}(\mathcal{M})=\{1^n(23)^n(45)^n6^n|n\geq 1\}.$$

The derivation steps performed by the above 1WK automaton, in order to accept the language L, are depicted below:

$$\underbrace{s_a\binom{a}{\lambda}s_a\ldots s_a\binom{a}{\lambda}}_{\text{n times}}\underbrace{s_a\binom{b}{\lambda}s_b\binom{\lambda}{a}s_a\ldots s_a\binom{b}{\lambda}s_b\binom{\lambda}{a}}_{\text{n times}}\underbrace{s_a\binom{c}{\lambda}s_c\binom{\lambda}{b}\ldots s_a\binom{c}{\lambda}s_c\binom{\lambda}{b}}_{\text{n times}}\underbrace{s_a\binom{\lambda}{c}s_a\ldots s_a\binom{\lambda}{c}s_a}_{\text{n times}}$$

The rpcfa(2), that simulates the above WK automaton is : $\mathcal{A}=(\{a,b,c\},A_1,A_2,K)$, where A_1 and A_2 are two finite automata with the initial and final states s_a and r_0, respectively, and the transition mappings defined as:

$$\begin{aligned}
&\bar{1}.\delta_1(s_a,a)=s_a, && \bar{5}.\delta_2(r_0,\lambda)=K_1\\
&\bar{2}.\delta_1(s_a,b)=s_b, && \bar{6}.\delta_2(s_a,\lambda)=K_1,\\
&\bar{3}.\delta_1(s_a,c)=s_c, && \bar{7}.\delta_2(s_b,a)=s_a,\\
&\bar{4}.\delta_1(s_a,\lambda)=K_2, && \bar{8}.\delta_2(s_c,b)=s_a,\\
& && \bar{9}.\delta_2(r_0,c)=s_a.
\end{aligned}$$

In the above system the WK rules are simulated as follows: 1 is simulated by $\bar{1}\bar{5}$ and $\bar{1}\bar{6}$, 2 is simulated by $\bar{2}\bar{6}$ and $\bar{2}\bar{5}$, 3 by $\bar{4}\bar{7}$, 4 by $\bar{3}\bar{5}$, 5 by $\bar{4}\bar{8}$ and 6 by $\bar{4}\bar{9}$. the controlled language of the $rpcfa(2)$ will be: $L_{ctr}(\mathcal{A})=\{\bar{1}\bar{5}(\bar{1}\bar{6})^{n-1}\bar{2}\bar{6}\bar{4}\bar{7}(\bar{2}\bar{5}\bar{4}\bar{7})^{n-1}(\bar{3}\bar{5}\bar{4}\bar{8})^n(\bar{4}\bar{9})^n|n\geq 1\}$.

For the rcpcfa(2), that simulates the above WK automaton, the δ_i functions are described in the following:

$$\begin{aligned}
&\bar{1}.\delta_1(s_a,a)=s_a, && \bar{5}.\delta_2(r_0,\lambda)=K_1\\
&\bar{2}.\delta_1(s_a,b)=s_b, && \bar{6}.\delta_2(s_a,\lambda)=K_1,\\
&\bar{3}.\delta_1(s_a,c)=s_c, && \bar{7}.\delta_2(s_b,a)=K_1,\\
&\bar{4}.\delta_1(s_a,\lambda)=s_a, && \bar{8}.\delta_2(s_c,b)=K_1,\\
& && \bar{9}.\delta_2(s_a,c)=K_1.
\end{aligned}$$

The simulation is: 1 is simulated by $\bar{1}\bar{5}$ and $\bar{1}\bar{6}$, 2 is by $\bar{2}\bar{6}$, 3 by $\bar{4}\bar{7}$, 4 by $\bar{3}\bar{6}$, 5 by $\bar{4}\bar{8}$ and 6 by $\bar{4}\bar{9}$. the controlled language of the $rcpcfa(2)$ is:

$$L_{ctr}(\mathcal{A})=\{\bar{1}\bar{5}(\bar{1}\bar{6})^{n-1}(\bar{2}\bar{6}\bar{4}\bar{7})^n(\bar{3}\bar{6}\bar{4}\bar{8})^n(\bar{4}\bar{9})^n|n\geq 1\}.$$

The cpcfa(2), that simulates the same WK automaton is:

$$\begin{aligned}
&\bar{1}.\delta_1(s_a,a)=s_a, && \bar{7}.\delta_2(r_0,\lambda)=K_1\\
&\bar{2}.\delta_1(s_a,b)=s_b, && \bar{8}.\delta_2(s_a,\lambda)=K_1,\\
&\bar{3}.\delta_1(s_a,c)=s_c, && \bar{9}.\delta_2(s_b,a)=K_1,\\
&\bar{4}.\delta_1(s_a,\lambda)=s_a, && \bar{1}\bar{0}.\delta_2(s_c,b)=K_1,\\
&\bar{5}.\delta_1(s_b,\lambda)=s_a, && \bar{1}\bar{1}.\delta_2(s_a,c)=K_1,\\
&\bar{6}.\delta_1(s_c,\lambda)=s_a. &&
\end{aligned}$$

In which 1 is simulated by $\bar{1}\bar{7}$ and $\bar{1}\bar{8}$, 2 is by $\bar{2}\bar{8}$, 3 by $\bar{5}\bar{9}$, 4 by $\bar{3}\bar{8}$, 5 by $\bar{6}\bar{1}\bar{0}$ and 6 by $\bar{4}\bar{1}\bar{1}$. the controlled language of the $pcfa(2)$ is:

$$L_{ctr}(\mathcal{A})=\{\bar{1}\bar{7}(\bar{1}\bar{8})^{n-1}(\bar{2}\bar{8}\bar{5}\bar{9})^n(\bar{3}\bar{8}\bar{6}\bar{1}\bar{0})^n(\bar{4}\bar{1}\bar{1})^n|n\geq 1\}.$$

The next results have been proven in [21]:

Theorem 2.
Each language $L \in RE$ can be written in the form $L = h(L')$, where $L' \in AWK(u)$ and h is a projection[7].

Theorem 3.
Each language $L \in RE$ is the image of a deterministic gsm[8] *mapping of a language in the $XWK(ctr)$, $X \in \{A, N, 1, 1N\}$.*

From theorem 1. and theorem 2, using the relation (9), we obtain:

Theorem 4.
Each language $L \in RE$ can be written in the form $L = h(L')$, where $L' \in X(2)$, with $X \in \{RPCFA, RCPCFA, CPCFA, PCFA\}$ and h a projection.

From Theorem 1. and Theorem 3. we get to:

Theorem 5.
Each language $L \in RE$ is the image of a deterministic gsm mapping of a language in the family $PCFA(2)(ctr)$.

3.2 From Watson-Crick automata to the computational power of pcft(2)

If an 1WK automaton is improved with the output capability the resulting device is known in the literature as Watson-Crick finite transducer (henceforth 1WKT). They are explicitly defined in [21]. In the previous section we have seen that Watson-Crick automata are powerful equivalent with $pcfa(2)$. Because of this the same equivalence is true for the corresponding transducers. As 1WKT are able to cover the family RE starting from regular languages it is expected that for PCFT(2) the same property holds. The next operations, on words and languages, are useful in the sequel:

The shuffle of two strings is an arbitrary interleaving of the substrings of two original strings. For two strings $x, y \in \Sigma^*$ and two symbols $a, b \in \Sigma$, we have:

(i) $x \sqcup \epsilon = \epsilon \sqcup x = x$, (ii) $ax \sqcup by = a(x \sqcup by) \cup b(ax \sqcup y)$.

The shuffle of two languages L_1, L_2 is defined as:

$$L_1 \sqcup L_2 = \bigcup_{x \in L_1, y \in L_2} x \sqcup y.$$

The twin shuffle language over the alphabet Σ, defined by:

$$TS_\Sigma = \bigcup_{x \in V^*} x \sqcup \bar{x}.$$

The following representation of the family RE is well known from [21]:

Theorem 6. *Each recursively enumerable language $L \subseteq T^*$ can be written as $L = h(TS_\Sigma \cap R)$, where Σ is an alphabet, R is a regular language, and h is a projection.*

Furthermore, in [21] it is proven that:

Lemma 2. *For every alphabet Σ, $TS_\Sigma \in N1WK(ctr)$.*

Lemma 3. $N1WK(ctr) \subset 1WK(ctr) \subset AWK(ctr) \subset CS$.

From Lemma 2. and Lemma 3, using the Corollary 1. we conclude that:

Lemma 4. *For every alphabet Σ, $TS_\Sigma \in PCFA(2)(ctr)$.*

As the controlled language of a $pcfa(2)$ can be expressed as the output of a $pcft(2)$, from Lemma 4. we get to the following result:

Corollary 2. *For every alphabet Σ, there is a $pcft(2)$, such that $TS_\Sigma = T(PCFA(2))$, where T is the corespond-ing $pcft(2)$ mapping.*

From Theorem 5. and Corollary 2. we obtain:

Corollary 3. *Each language $L \in RE$ can be written in the form $L = g(T(PCFA(2)))$, where g is a deterministic gsm and T is the corresponding $pcft(2)$ mapping.*

---

[7] A homomorphism that erases some symbols and leaves unchanged the others.

[8] A *generalized sequential machine (sequential transducer)* is a system $g = (Q, \Sigma, \Delta, q_0, F, \delta)$, where Σ and Δ are alphabets (the input and the output alphabet respectively), Q is the set of states, q_0 is the initial state, F is the set of final states, and $\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q \times \Delta^*)$.

From Theorem 6 and Corollary 2, taking into consideration that both, intersection with regular languages and morphism are realized by pcft(2)[9], we get to:

Theorem 7. *For each recursively enumerable language $L \subseteq \Sigma^*$ there is a pcft(2), such that $L = T(PCFA(2))$, where T is the corresponding PCFT mapping.*

As $REG \subset 1WK = PCFA(2)$, see the relation (9), we conclude that PCFTS are able to cover the family of all RE languages having as input regular languages.

3.3 On the computational power of PCFPT

In order to express the computational power of PCFPT we will take into consideration only some constructive example. They come to show that the questions from the beginning of these section have a positive answer for these systems, too.

Example 2. Let us consider the next deterministic PCPT(2) :

$$\mathcal{T} = (\{a, c\}, \{Z_0, Z_1, a\}, T_1, T_2, \{K_1, K_2\}),$$

in which $T_i = (Q_i, \Sigma, \Gamma, \Delta, \delta_i, q_i, Z_i, F_i)$, $i \in \{1, 2\}$, are two pushdown automata, defined as:

$$T_1 = (\{q_1, s_1, r_1, p_1\}, \{a, c\}, \{Z_0, Z_1, a\}, \{a\}, \delta_1, q_1, Z_1, \{p_1\}),$$
$$T_2 = (\{q_2, s_2, p_2\}, \{a, c\}, \{Z_0, Z_1, a\}, \{a\}, \delta_2, q_2, Z_1, \{p_2\}),$$

having the next transition mappings :

1. $\delta_1(q_1, a, Z_1) = (s_1, Z_1, a)$
2. $\delta_1(s_1, a, Z_1) = (s_1, aaZ_1, a)$
3. $\delta_1(s_1, a, a) = (s_1, aaa, a)$
4. $\delta_1(s_1, c, a) = (r_1, Z_0aaa, \lambda)$
5. $\delta_1(r_1, \lambda, a) = (r_1, \lambda, \lambda)$
6. $\delta_1(r_1, \lambda, Z_0) = (r_1, \lambda, \lambda)$
7. $\delta_1(r_1, \lambda, Z_1) = (p_1, K_2, a)$
8. $\delta_1(p_1, \lambda, a) = (p_1, \lambda, a)$
9. $\delta_2(q_2, \lambda, Z_1) = (q_2, K_1, \lambda)$
10. $\delta_2(q_2, \lambda, a) = (q_2, K_1a, a)$
11. $\delta_2(q_2, \lambda, Z_0) = (s_2, \lambda, a)$
12. $\delta_2(s_2, a, a) = (s_2, \lambda, \lambda)$
13. $\delta_2(s_2, c, a) = (p_2, \lambda, \lambda)$
14. $\delta_2(p_2, \lambda, a) = (p_2, \lambda, \lambda)$
15. $\delta_2(p_2, \lambda, Z_1) = (r_2, \lambda, \lambda)$
16. $\delta_2(r_2, \lambda, a) = (r_2, \lambda, a)$
17. $\delta_2(r_2, \lambda, Z_1) = (r_2, \lambda, \lambda)$

Taking as input, the regular language: $L = \{a^n c \mid n \geq 1\}$ the PCPTS will generate the following context-sensitive language $L = \{a^{n^2} \mid n \geq 1\}$, by following the transitions: $\underline{(q_1, a^n c, Z_1, \lambda, q_2, a^n c, Z_1, \lambda)}$

$\vdash_{1+9} (s_1, a^{n-1}c, Z_1, a, q_2, a^n c, K_1, \lambda)$

$\vdash_{2+9} (s_1, a^{n-2}c, aaZ_1, aa, q_2, a^n c, K_1, \lambda)$

$\vdash^{*(n-2)}_{3+10} (s_1, c, a^{2(n-1)}Z_1, a^n, q_2, a^n c, a^{2(n-1)}Z_1 ... a^2 Z_1, a^{n-2})$

$\vdash_{4+10} (r_1, \lambda, Z_0 a^{2n} Z_1, a^n, q_2, a^n c, K_1 a^{2(n-1)} Z_1 .. a^2 Z_1, a^{n-1})$

$\vdash_{6+11} (r_1, \lambda, a^{2n} Z_1, a^n, s_2, a^n c, a^{2n} Z_1 a^{2(n-1)} Z_1 .. a^2 Z_1, a^n)$

$\vdash^{*n}_{5+12} (r_1, \lambda, a^n Z_1, a^n, s_2, c, a^n Z_1 a^{2(n-1)} Z_1 .. a^2 Z_1, a^n)$

$\vdash_{5+13} (r_1, \lambda, a^{(n-1)} Z_1, a^n, p_2, \lambda, a^{(n-1)} Z_1 a^{2(n-1)} Z_1 .. a^2 Z_1, a^n)$

$\vdash^{*(n-1)}_{5+14} (r_1, \lambda, Z_1, a^n, p_2, \lambda, Z_1 a^{2(n-1)} .. a^2 Z_1, a^n)$

$\vdash_{7+15} (p_1, \lambda, K_2, a^n, r_2, \lambda, a^{2(n-1)} Z_1 ... a^2 Z_1, a^n)$

$\vdash^{*2(n-1)}_{8+16} (p_1, \lambda, Z_1 ... a^2 Z_1, a^n a^{2(n-1)}, p_2, \lambda, Z_1 a ... a^2 Z_1, a^n a^{2(n-1)})$

$\vdash_{7+17} (p_1, \lambda, a^{2(n-2)} Z_1 .. a^2 Z_1, a^n a^{2(n-1)}, p_2, \lambda, a^{2(n-2)} Z_1 .. a^2 Z_1, a^n a^{2(n-1)})$

$\vdash^{*2(n-2)}_{8+16} (p_1, \lambda, Z_1 a^{2(n-3)} a^2 Z_1, a^n a^{2(n-1)} a^{2(n-2)}, p_2, \lambda, Z_1 a^{2(n-3)} a^2 Z_1, a^n a^{2(n-1)} a^{2(n-2)})$

$\vdash_{7+17} ... \vdash^{*2(n-3)}_{8+16} ... \vdash_{7+17} ... \vdash^{*2(n-4)}_{8+16} ... \vdash_{7+17} ... \vdash^{*4}_{8+16} ... \vdash_{7+17} \vdash^{*2}_{8+16} ...$

$\vdash_{7+17} \underline{(p_1, \lambda, \lambda, a^n a^{2(n-1)} a^{2(n-2)} a^{2(n-3)} .. a^4 a^2, p_2, \lambda, \lambda, a^n a^{2(n-1)} a^{2(n-2)} a^{2(n-3)} .. a^4 a^2)}$

The output is: $a^n a^{2(n-1)} a^{2(n-2)} \ldots a^4 a^2 = a^{2n+2(n-1)+...+4+2-n} = a^{n(n+1)-n} = a^{n^2}$.

The meaning is that there exist context-sensitive languages, that can be generated by PCPTS, taking as input a regular language. Furthermore, the next result holds:

Theorem 8.

For each recursively enumerable language L_{RE} there exist a regular language L_{REG} and a PCPTS such that $T(L_{REG}) = L_{RE}$.

[9] 1WKT are able to perform these.

We leave open the problem of computational power of PCPTS, since we have another example that stands for NLP.

Example 3. We will build a PCPTS in order to check the correctness of the next grammatical structure: **pronoun–clitic**[10]**–verb**. We want also, to simulate phonological phenomena that appear in such a phrase. Let $\mathcal{A}=(\Sigma,\Gamma,T_1,T_2,T_3,K)$ be an EPCPT, where $\Sigma=\Sigma_R\cup\{1,2,3,\check{s},\check{p},\sharp\}$, Σ_R being the Romanian alphabet. We consider as input, strings of the form: **eu** $\sharp 1\check{s}$ **îmi** $\sharp 1\check{s}$**întîlnesc**$\sharp 1\check{s}$, where the first symbol behind $\sharp$ stands for person and the second for number ($\check{s}$ is for singular and $\check{p}$ is for plural). If all the grammatical agreements hold, the output must be: **eu mi-ntîlnesc** (I am meeting my...). The δ rules are defined as being:

$\delta_i(q_i,x,Z_i)=(q_i,Z_i,x)$, where $x\in X_1=\{a,e,i,l,o,n,t,u,v\}$, $i\in\{1,2,3\}$;

The functions read and output the personal pronoun.

$\delta_i(q_i,\sharp,Z_i)=(s_i,Z_i,\lambda)$, for all $i\in\{1,2,3\}$;

The symbol $\sharp$ changes the states, in order to obtain a deterministic system.

$\delta_1(s_1,x,Z_1)=(s_1,xZ_1,\lambda)$, where $x\in X_2$, $X_2=\{1,2,3\}$;

$\delta_1(s_1,y,x)=(s_1,xy,\lambda)$, where $x\in X_2$ and y_3, $X_3=\{\check{s},\check{p}\}$;

$\delta_i(s_i,x,Z_i)=(s_i,Z_i,\lambda)$, for $i\in\{2,3\}$ and $x\in X_2\cup X_3$;

Only the first stack stores the grammatical attributes: the person and the number.

$\delta_1(s_1,x,z)=(s_1,z,\lambda)$, where $x\in\Sigma-X_3$ and $z\in X_2$;

$\delta_3(s_3,x,Z_3)=(s_3,Z_3,\lambda)$, $x\in\Sigma-X_3$; $\delta_2(s_2,\hat{\imath},Z_2)=(s_2,\hat{\imath}Z_2,\lambda)$;

$\delta_2(s_2,x,\hat{\imath})=(s_2,\hat{\imath}x,\lambda)$, where x $\in$ {m, l , ţ , ş};

$\delta_2(s_2,i,\{\hat{\imath}m,\hat{\imath}ţ,\hat{\imath}ş\})=(s_2,\hat{\imath}\{m,ţ,ş\}i,\lambda)$, $\delta_2(s_2,\sharp,\hat{\imath})=(s_2,K_1\,\hat{\imath},\lambda)$;

Clitic pronouns will only be read, but the second stack memorizes them. The second stack asks also the first one, to see if the clitic agrees in number and person with the pronoun. The spelling is made using the Romanian grammar rules [11].

$\delta_2(s_2,1,\{2,3\})=(s_2,\lambda,\lambda)$, $\delta_2(s_2,2,\{1,3\})=(s_2,\lambda,\lambda)$;

$\delta_2(s_2,3,\{1,2,3\})=(s_2,\lambda,\lambda)$, $\delta_2(s_2,x,x)=(s_2,x,\lambda)$, $x\in X_2-\{3\}$;

$\delta_2(s_2,y,xy)=(p_2,\lambda,\lambda)$, $x\in X_2-\{3\}$, $y\in X_3$, $\delta_2(p_2,\lambda,y)=(r_2,\lambda,\lambda)$, $y\in X_3$;

Other situations will block the system, the functions not being defined[12].

$\delta_1(s_1,X_3,X_2)=(p_1,K_2,\lambda)$, $\delta_3(s_3,X_3,Z_3)=(p_3,K_2,\lambda)$, $\delta_2(s_2,X_3,X_3)=(r_2,\lambda,\lambda)$;

The first and the third transducer ask the second one for the memorized clitic.

$\delta_i(p_i,\lambda,X_3)=(r_i,\lambda,\lambda)$, for all $i\in\{1,2,3\}$;

$\delta_i(r_i,\{\hat{\imath},\hat{\imath}\},Z_1\{\hat{\imath}mi,\hat{\imath}ţi,\hat{\imath}şi\})=(r_i,\lambda,\{$ mi-, ţi-, şi- $\})$, respectively, $i\in\{1,2,3\}$;

$\delta_i(r_i,a,Z_1\{\hat{\imath}mi,\hat{\imath}ţi,\hat{\imath}şi\})=(r_i,\lambda,\{$ mi-a, ţi-a, şi-a $\})$, $i\in\{1,2,3\}$;

$\delta_i(r_i,x,Z_1\{\hat{\imath}mi,\hat{\imath}ţi,\hat{\imath}şi\})=(r_i,\lambda,\{$ îmi, îţi, îşi $\})$, $i\in\{1,2,3\}$, $x\notin\{a,i,\hat{\imath}\}$;

All the components output the clitic pronoun.

$\delta_i(r_i,\Sigma_R,\{$ îmi, îţi, îşi $\}Z_2)=(r_1,\lambda,\Sigma_R)\}$, for all $i\in\{1,2,3\}$;

The content of each stack is deleted until it gets to the pronoun features. In the same time all the components output the verb that follows the clitic.

$\delta_1(r_1,\Sigma_R,X_2)=(r_1,X_2,\Sigma_R)$; $\delta_1(r_1,x,X_2)=(r_1,X_2,\lambda)$, x $\in X_2\cup X_3\cup\{\sharp\}$;

$\delta_2(r_2,\Sigma_R,\lambda)=(r_2,\lambda,\Sigma_R)$; $\delta_2(r_2,x,\lambda)=(r_2,\lambda,\lambda)$, x $\in X_2\cup X_3\cup\{\sharp\}$;

$\delta_3(r_3,\Sigma_R,Z_3)=(r_3,Z_3,\Sigma_R)$, $\delta_3(r_3,\sharp,Z_3)=(r_3,K_1,\lambda)$;

After all the components read and output the verb, the third stack asks the first one, in order to check if the verb and the pronoun agree in number and person:

$\delta_3(r_3,x,x)=(r_3,\lambda,\lambda)$, x $\in X_2$; $\delta_3(r_3,y,y)=(r_3,\lambda,\lambda)$, y $\in X_3$;

Taking r_i as final state for the i-th automaton, $1\le i\le 3$, in the case that all the agreements are fulfilled, the system has as output strings such as: **eu mi-ntîlnesc** for the input: **eu îmi întîlnesc** (*I am meeting my..*), or **el şi - aduce** for **el îşi aduce** (*He is bringing to him..*),

[10] A clitic, in the Romanian language, is an unstressed pronoun.

[11] Rules such as concordance between pronoun and verb, or between strong pronoun and clitic are used.

[12] The Romanian clitic system does not allow, for instance, combinations such as:1$\check{s}$ (pers.1 singular) for pronoun and 1$\check{p}$ (pers.1 plural) for clitic. A pair such as:(1$\check{s}$,1$\check{p}$) will be discarded.

and so on. These operations are known in phonology as vowel deletion phenomena.

4 Conclusions

In this paper we have introduced the notion of PCTS and other definitions related to this topic. We dealt with the mechanism of parallel communication and we have applied it in the natural language processing. A description of Watson-Crick automata has been done in order to describe the computational power of pcft(2). We have touched the problem of PCPTS taking into consideration various examples. Several characterisations of recursively enumerable languages have been given for these systems.

References

[1] A. O. Buda.1977. Multiprocessor automata, Inform. Proces. Lett. 25(1977),257-261.

[2] E. Csuhaj-Varju, C. Martin-Vide, V. Mitrana, G. Vaszil. Parallel communicating pushdown automata systems. Internat. J. Found. Comput. Sci. 11 (2000), no. 4, 633-650.

[3] J. Dassow, V. Mitrana, Stack cooperation in multi-stack pushdown automata, J. Comput. System Sci. 58(1999), 611-621.

[4] R. J. Evey. 1963. The Theory and Application of Pushdown Store Machines, Ph.D. Thesis and Research Report, Mathe. Linguistics and Automat. Trans. Project, Harvard University, NSF-10, May1963.

[5] P. C. Ficher.1963. On Computability by certain classes of restricted Turing machines, Proc.Fourth IEEE Symp. on Switch. Circuit Theory and Logical Design.

[6] S. Ginsburg. 1962. Example of abstract machines, IEEE Trans. on Electronic Computers 111: 2, 132-135.

[7] S. Ginsburg, G. F. Rose. 1966. it Preservation of languages by transducers, Information and Control 9:2, 153-176.

[8] O. H. Ibarra. 1973. One two-way multihead automata, J. Comput. System Sci. 7(1973), 28-36.

[9] R. M. Kaplan, M. Kay. 1994. Regular models of phonological rule systems, Computational Linguistics, 20(3)

[10] L. Karttunen. 1993. Finite-state lexicon compiler, Technical Report Xerox PARC P93-00077, Xerox PARC.

[11] W. Kuich, A. Salomaa. 1986. Semirings, Automata, Languages, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin, Germany.

[12] M. D. Jiménez-Lopéz, C. Martín-Vide. 1997.Grammar Systems for the Description of Certain Natural Language, New Trends in Formal Languages, Contreol, Cooperation and Combinatorics. Gheorghe Pãun and Arto Salomaa (Eds.)

[13] C. Martín-Vide, V. Mitrana.1997. Parallel Communicating Automata Systems, A Survey.Korean J. Comput. Appl. Math. 7 (2000), no. 2, 237-257.

[14] C. Martín-Vide, V. Mitrana, Some undecidable problems for parallel communicating finite automata systems. Inform. Process. Lett. 77 (2001), no. 5-6, 239-245.

[15] C. Martín-Vide, A. Mateescu, V. Mitrana. Parallel finite automata systems communicating by states. International Journal of Foundation of Computer Science Vol. 13 No. 5 (2002) 733-749.

[16] V. Mitrana. 1999. On the degree of communication in parallel communicating finite automata systems. Descriptional complexity of automata, grammars and related structures (Magdeburg, 1999). J. Autom. Lang. Comb. 5 (2000), no. 3, 301-314.

[17] M. Mohri. 1997. Finite-State Transducers in Language and Speech Processing, Computational Linguistics, 23(2), pp. 269-311.

[18] M. Mohri. 2000. Minimization algorithms for sequential transducers, Theoretical Computer Science, 234:177201.

[19] M. Mohri, F. Pereira, M. Riley. 1996. Weighted automata in text and speech processing, In ECAI-96 Workshop, Budapest, Hungary. ECAI.

[20] G. van Noord, D. Gerdemann. 2001. Finite State Transducers with Predicates and Identities.

[21] G. Paun, G. Rozenberg, A. Salomaa. 1998. DNA Computing: new Computing Paradigms.